AMENDMENTS TO THE SPECIFICATION:

Please amend the specification as follows:

On page 1, after the title and before line 4, please add the following paragraph:

This application is a national stage filing under 35 U.S.C. § 371 of International Application No. PCT/EP2003/009832, filed on September 4, 2003, which published in the English language, and claims the benefit of priority to U.S. Provisional Application Nos. 60/408,901, filed on September 9, 2002, 60/408,902, filed on September 9, 2002, 60/408,903, filed on September 9, 2002, 60/408,905, filed on September 9, 2002, 60/409,606, filed on September 11, 2002, and 60/409,593, filed on September 11, 2002.

Page 2, please amend the paragraph beginning at line 24, as follows:

As long as the data objects are still available in the original ~~data base~~ database, they can still be modified during ~~said~~ the time gap. Because the deleting program does not compare the archived data object and the data object to be deleted, such modifications can be lost. This has not only the consequence of the loss of the amended data, it can additionally have the consequence that certain business processes can not be completed.

Page 2, please amend the paragraph beginning at line 33, as follows:

~~An other~~ Another problem arises[[,]] if several archiving processes run in parallel. ~~Then it can happen, that~~ In this scenario, one data object ~~is~~ can be archived several times, and ~~is~~ no longer unambiguously identifiable. This can have the consequence

that evaluations or statistical analysis, which use the archive files, produce wrong results.

Page 3, please amend the paragraph beginning at line 5, as follows:

It ~~can also happen~~ is also possible that data objects in the original ~~data base~~ database are read by the writing module and are simultaneously modified by ~~an other~~ another software application. In such a case, the data can be transferred from an archiveable status to a ~~non archiveable~~ non-archiveable status. ~~In consequence~~ As a result, data objects which are not archiveable are written into the archive file and are deleted from the original ~~data base~~ database. In effect, this can result in a loss of data.

Page 3, please amend the paragraph beginning at line 19, as follows:

In accordance with one embodiment of the invention, as embodied and broadly described herein, methods and systems consistent with the principles of the invention provide for moving data objects in a computer system from a first storage location to a second storage location, comprising: [[

a)]] selecting one or more data objects having an identifier (ID) from the first storage location[[,]]; [[

b)]] storing [[said]] the ID in a first lock object[[,]]; [[

c)]] storing [[said]] the ID in a second lock object[[,]]; [[

d)]] storing a data object, the ID of which is contained in the first lock object, at the second storage location[[,]]; [[

e)]] deleting ~~a~~ the data object, the ID of which is contained in the first lock object, from [[said]] the first storage location[[,]]; [[

f)]] deleting ~~an~~ the ID from the first lock object ~~earliest at a time at which step c)~~ after the step of storing the ID in the second lock object for the respective data object assigned to that ID has been completed;[[,]] and [[

g)]] deleting ~~an~~ the ID from the second lock object ~~earliest at a time at which step b)~~ after the step of storing the ID in the first lock object for a particular ID has been completed.

Page 4, please amend the paragraph beginning at line 15, as follows:

In accordance with another aspect[[,]] of the invention, as embodied and broadly described herein, methods and systems consistent with the principles of the invention provide a computer system for processing data by means of or in a software application, comprising:

[[-]] memory means for storing program instructions;

[[-]]input means for entering data;

[[-]]storage means for storing data;

[[-]]a processor responsive to program instructions; and

[[-]]program instructions adapted to carry out [[a]] the method ~~as of any of claims 1 to 12~~ described above.

Page 4, please amend the paragraph beginning at line 32, as follows:

~~An~~One advantage of the invention and its embodiments is that the security against data loss in data moving and archiving procedures ~~is~~ may be greatly improved. This ~~avoids in consequence~~ may avoid spending a lot of time and/or money for data ~~retrieving~~ retrieval.

Page 5, please amend the paragraph beginning at line 25, as follows:

Fig. 1 is a schematic block diagram of ~~the~~ an exemplary ~~implementation of the inventive method within a~~ computer system for implementing methods consistent with the present invention.

Page 6, please amend the paragraph beginning at line 1, as follows:

Fig. 3 is ~~an exemplary~~ a flow diagram of an exemplary implementation of the selecting module shown in Fig. 1.

Page 6, please amend the paragraph beginning at line 4, as follows:

Fig. 4 is ~~an exemplary~~ a flow diagram of an exemplary method for implementation [[of]]by the writing module shown in Fig. 1.

Page 6, please amend the paragraph beginning at line 7, as follows:

Fig. 5 is ~~an exemplary~~ a flow diagram of an exemplary method for implementation [[of]]by the deleting module shown in Fig. 1.

Page 6, please amend the paragraph beginning at line 10, as follows:

Fig. 6 is ~~an exemplary~~ a flow chart of ~~a further~~ an exemplary method for implementation [[of]]by the selection and writing module [[mentioned]]shown in Fig. 1.

Page 6, please amend the paragraph beginning at line 14, as follows:

Fig. 7 ~~shows an exemplary~~ is a flow chart of an exemplary method to demonstrate how any software application ~~can~~ may use the concept of the P[[-]] and T-locks.

Page 6, please amend the paragraph beginning at line 18, as follows:

Fig. 8 ~~shows a process alternative to that shown in Fig. 7~~ is a flow chart of an exemplary method to demonstrate how any software application may use the concept of the P and T-locks, including a conditional deletion of a P-lock.

Page 6, please amend the paragraph beginning at line 21, as follows:

Fig. 9 ~~shows an example of~~ is a flow chart for an exemplary method for implementation by a software module ~~by means of~~ through which the locks can be deleted.

Page 6, please amend the paragraph beginning at line 25, as follows:

Computer ~~system~~ systems and ~~program~~ programs are closely related. As used hereinafter, phrases, such as "the computer provides," ~~and~~ "the program provides or performs specific actions[[",]]," and "a user performs a specific action" are convenient ~~abbreviation~~ abbreviations to express actions by a computer system that is controlled by a program or to express that the program or program module is designed to enable the computer system to perform the specific action or the enable a user to perform the specific action by means of a computer system.

Page 8, please amend the paragraph beginning at line 15, as follows:

An identifier (ID) is a type of data, which allows an unambiguous identification of the data object to be archived[[,]]. ~~it~~ It can be implemented, for example, as a number or a combination of alphanumerical characters or as a characteristic part of the data object to be archived. It is clear from that definition that a data object can have a wide

variety of IDs. A lock object is a data object, in which the identifiers are stored. It can be implemented, e.g., as a file on a storage means or as a data array in computer memory. The first lock object is stored advantageously in a nonvolatile storage means. The second lock object can be stored in volatile and/or nonvolatile storage means.

Page 8, please amend the paragraph beginning at line 29, as follows:

Fig. 1 ~~depicts one example~~ is a schematic block diagram of an exemplary implementation of a computer system ~~a first embodiment of the invention~~. Fig. 1 shows a computer system 101 comprising a computer 103 having a CPU 105, a working storage 112, in which an ERP software 111 is stored for ~~being processed~~ processing by CPU 105. The second lock object is stored in working storage 112 as well. ERP software 111 comprises program modules 106, 109, 110 for carrying out the inventive data archiving process. Computer [[S]]]system 101 further comprises input means 113, output means 112 for interaction with a user, and general input/output means 104, including a net connection 114, for sending and receiving data. A plurality of computer systems 101 can be connected via the net connection 114 in the form of a network 113. In this case, the network computers 113 can be used as further input/output means, including the use as further storage locations. Computer system 103 further comprises a first storage means 107, in which data to be archived and the first lock object are stored, and a second storage means 108, in which the archived data are stored.

Page 10, please amend the paragraph beginning at line 1, as follows:

In an alternative embodiment, the lock object ~~is~~ may be created by the selection module and not by the writing module.

Page 10, please amend the paragraph beginning at line 5, as follows:

In a second implementation of the invention, a data object to be archived ~~comprises~~ may comprise of one or more fields of one or more tables, and the ID of the respective object ~~comprises~~ may comprise of one or more key fields of that data object. This can best be seen from Fig. 2. In this instance, various sets of data objects are created in the form of two-dimensional data arrays, i.e., two tables ~~201, 202~~ (table 1 and table 2) having columns named field A to field X and field A to field Y, respectively, and a certain, unspecified number of lines. A field of the array or table is defined by the name of the column and the respective line. Such a field can contain data to be archived. It can alternatively contain a reference to a line of a further table. For example, in table 1 field X in line 2 contains a reference to line 3 in table 2. A data object 201.x to be archived comprises of fields of one line of the respective table. If one of the fields contains a reference to a line of an other table, fields of this referenced line belong to the data object~~, too~~ as well. In the example in Fig. 2, a data object 201.x to be archived comprises the fields of line 2 in table 1 and fields of line 3 in table 2.

Page 10, please amend the paragraph beginning at line 27, as follows:

An ID of such a data object can be implemented by the content of one or more so-called key fields, if the combination of these key fields is unique within the respective table. In the example, the fields of "field A" and "field B" can be used as key fields for table 1, whereas field A alone is key field in table 2. Within this example, the data object 201.x has the content of the fields of columns field A and B of the respective lines as ID. The ID for the data object 201.x to be archived is stored as a first type ID in the

first lock object 203, named permanent lock object in Fig. 2, and as a second type ID in the second lock object 204, named transactional lock object. The permanent lock object [[is]]may be implemented as a table having two columns, the first of which contains the ID. The second type lock object 204 can be implemented as a one-dimensional data array stored in the working memory of the computer system. However, it can be implemented as file on a nonvolatile storage means, too. The first type ID, ID 1, is deleted after the selected data object 201.x has been deleted according to ~~step e) of~~ the inventive process, and the second type ID, ID 2, ~~is~~ may be deleted immediately after ~~the time as defined in step g)~~. Alternatively, type ID 1 IDs can be deleted after all the selected data objects have been deleted ~~according to step e)~~. As can be seen, both ID types have identical content, the ID of the respective lines of the data to be archived. However, this is not a necessary condition. Different contents can be used for the different ID types. The permanent lock objects further contain a column by which a filename ~~is~~ may be assigned to the ID of the data object, i.e., that data object to be archived. In the example, line 1 is archived in a file named 001, lines 2 and 3 in file 002, and line 4 in file 003.

Page 12, please amend the paragraph beginning at line 1, as follows:

A further embodiment ~~is characterized in that in step c) the ID is stored~~ may comprise storing the ID in the second lock object ~~in step c)~~ immediately after ~~performing step a)~~ selecting one or more data objects having an identifier (ID) from the first storage location for the respective data object. Alternatively, ~~in step c)~~ the ID of the selected

data object is stored in the second lock object ~~shortly~~ before ~~the~~ storing ~~process according to step d) for~~ the data object assigned to that ID is started.

Page 12, please amend the paragraph beginning at line 11, as follows:

A further embodiment ~~is characterized in that in step b)~~ may comprise storing the IDs of all selected data objects ~~are stored in step b)~~ in the first lock object before ~~the first~~ storing the data object at the second storage location ~~process according to step d) is started~~.

Page 12, please amend the paragraph beginning at line 16, as follows:

In a further embodiment the invention ~~comprises~~ may comprise ~~h)~~ checking ~~before or while performing any of steps a) to c) for a data object,~~ whether an ID for the data object has been stored in a first lock object, and if the ID has been stored ~~yes~~, not storing the data object, the ID of which is contained in the first lock object, at the second storage location for that data object.

Page 12, please amend the paragraph beginning at line 22, as follows:

Additionally, the invention ~~comprises~~ may comprise ~~i)~~ checking ~~before or while performing any of steps a) to d) for a data object,~~ whether that data object is contained in the second storage location, and if ~~yes~~ the data object is contained in the second storage location, ~~skipping at least step d)~~ not storing the data object, the ID of which is contained in the first lock object, at the second storage location for that data object.

Page 12, please amend the paragraph beginning at line 28, as follows:

~~An other~~ Another embodiment ~~is characterized by said~~ may comprise checking ~~is performed~~ by querying a first lock object. Further, a method may be provided that comprises:

~~j) in case of a failure in step d)~~ determining whether the data object was stored in the first lock object successfully, and upon an unsuccessful storage, checking, whether the data object assigned to the respective ID has been completely stored in the second storage location, and ~~in case of no~~ if the respective ID has not been completely stored, skipping ~~at least~~ the step of deleting the data object from the first storage location and the step of deleting the ID from the first lock object after the respective data object assigned to that ID has been deleted ~~steps e) and f)~~ for that data object and deleting the ID from the first lock object.

Page 13, please amend the paragraph beginning at line 4, as follows:

Embodiments of [[The]] the invention [[is]] are now described in more detail with reference to Figs. 3 to 5, which are schematic flow diagrams of exemplary methods that may be implemented by ~~implementations of~~ the selecting, writing and deleting modules, respectively, as shown in Fig. 1. Within the context of this description, and particularly ~~the~~ with respect to Figs. 3 to 9, a first type ID is called a P-lock (permanent) and a second type ID is called a T-lock (transactional). ~~So~~ Therefore, setting a P- or T-lock for a selected object means to store an ID of that object in a respective lock object. The term "permanent" results from the property of the P-lock of existing permanently, as long as the data object is not yet deleted from its original storage location. The term

"transactional" results from the property of the T-lock of existing only as long as a specific action (e.g., checking of archiveability) is performed on a selected data object or, in other words, of being deleted ~~shortly~~ after the respective action has been performed.

Page 13, please amend the paragraph beginning at line 23, as follows:

In the exemplary flow chart of the selecting module in Fig. 3, a data object is selected in a first step 301. Subsequently, a T-lock is set on this object in a second step 302. If the T-lock was successfully set (step 303), that is, if it did not yet exist, it is checked in step 304 whether a P-lock already exists in the selected data object. If the T-Lock could not be set successfully, the next data object is selected (step 309). The setting of the T-lock (step 302) and the check (step 303), whether it is successfully set, can advantageously be implemented as one "atomic" step. This means that both steps can be executed essentially at the same time or, in other words, the time gap between both steps can be essentially zero.

Page 14, please amend the paragraph beginning at line 4, as follows:

Both checks (steps 303 and 304) can also be implemented by querying the respective lock objects. If a P-lock exists, the T-lock is deleted (step 308) and the next data object is selected (step 309). If no P-lock exists, it is checked in steps 305 and 306[[,]] whether the data object is archiveable. Such checking comprises a test of whether the data in the data object is readable, complete, or not fraught with obvious failures, etc. If the test is successful, a P-lock is set on that data object in step 307,

whereby no archive file is assigned to the data object. Then the T-lock is deleted (step 308) and the next data object is selected (step 309).

Page 15, please amend the paragraph beginning at line 8, as follows:

In the exemplary flow chart of a further exemplary implementation in Fig. 6, the selecting and writing module described above are combined to one module. Accordingly, a data object is selected in a first step 601. Subsequently, a T-lock is set on this object in step 602. If the T-lock was successfully set (step 603), it is checked in step 604 whether a P-lock already exists in the selected data object. If not, the next data object is selected (step 611). If a P-lock exists on that object, the T-lock is deleted (step 610) and the next data object is selected (step 611). If no P-lock exists on that object, it is checked in step 605[[,]] whether the data object is archiveable. If this check fails (step 606), the T-lock is deleted (step 610), and the next data object is selected (step 611). If the check (606) is positive, a P-Lock is set (607), the data object is stored (step 608) in an archive file, the archive file is assigned to the P-Lock (609), the T-lock is deleted (step 610), and the next data object is selected (step 611).

Page 15, please amend the paragraph beginning at line 29, as follows:

Fig. 7 shows ~~by way of an exemplary~~ a flow chart of an exemplary method to demonstrate how any software application can use the concept of the P[[-]] and T-locks to ensure that the measures, that the software application is going to apply on the data object, do not influence the archiving process. A software application which is programmed to have a read and/or write access to data objects, which can be subject of an archiving process as described, ~~comprises~~ may comprise the following steps as

shown in Fig. 7. In a first step 701, the data object is selected. Then, a T-lock is set in step 702 on that object by the application. If the T-lock is successfully set (step 703), it is checked in (step 704), whether a P-lock exists on that object[[,]]; otherwise the application terminates (step 707). If a P-lock exists on that object (step 704), the T-lock is deleted (step 706), and the application terminates (step 707). If no P-lock exists, i.e., the data object is not subject to an archiving process, the application can have read/write access to the data object in a working step 705. Subsequently, the application deletes the T-lock (step 706) and terminates (step 707).

Page 16, please amend the paragraph beginning at line 16, as follows:

Fig. 8 ~~shows a process alternative to that shown in Fig. 7~~ is a flow chart of another exemplary method to demonstrate how any software application may use the concept of the P and T-locks, including a conditional deletion of a P-lock. In a first step 801, the data object is selected. Then, a T-lock is set on that object by the application (step 802). If the T-lock is successfully set (step 803), it is checked (step 804)[[,]] whether a P-lock exists on that object[[,]]; otherwise the application terminates (step 809). If no P-lock exists (step 804), i.e., the data object is not subject to an archiving process, the application can have read/write access to the data object in working step 807. Subsequently, the application deletes the T-lock (step 808) and terminates (step 809). If a P-lock exists (step 804), it is checked (step 805)[[,]] whether a file is assigned to it. If a file is assigned, the application deletes the T-lock (step 808) and terminates (step 809). If no file is assigned, the P-lock is deleted (step 806), and the application

can have read/write access to the data object (step 807). Subsequently, the application deletes the T-lock (step 808) and terminates (step 809).

Page 17, please amend the paragraph beginning at line 8, as follows:

Fig. 9 ~~shows an example of~~ is a flow chart ~~for~~ of an exemplary method for implementation by a software module ~~by means of~~ through which the locks set by the modules described above can be deleted. This can be useful in cases in which no archive files are assigned to P-locks or in which P-locks have been deleted for a user. Therein, a P-lock is nothing else than a data object and can be treated in the same way as described above. In a first step 901, a P-lock is selected. Then, a T-lock is set to the P-lock in step 902. If the T-lock is successfully set (step 903), it is checked in step 904, whether the P-lock has a file assigned. If the T-lock is not set successfully, the module terminates (step 907). If the selected P-lock has no file assigned (step 904), the P-lock is deleted (step 905). Then, the T-lock is deleted (step 906), and the module terminates (step 907). Alternative to the termination (step 907), a next P-lock can be selected.